INFORMATION AND COMPUTATION 80, 165–191 (1989)

A Language Independent Proof of the Soundness and Completeness of Generalized Hoare Logic

PATRICK COUSOT

École Polytechnique, Laboratoire d'Informatique (LIX), 91128 Palaiseau – cedex, France

AND

RADHIA COUSOT

Université de Paris-Sud, Centre d'Orsay, LRI, Bat. 490, 91405 Orsay – cedex, France

Generalized Hoare logic (GHL) is a formal logical system for proving invariance properties of programs. It was first introduced by Lamport to reason about simple concurrent programs with shared variables. It was generalized by Lamport and Schneider who noticed that the inference rules for each language construct (enabling invariance properties of statements to be derived from invariance properties of their components) can be viewed as special cases of simple logical meta-rules. We give a rigorous definition of GHL, based upon an abstract formalization of the syntax and semantics of programs so as to provide an interpretation for formulas of GHL which is independent of the specific instructions of the programming language used. We prove that the proof system of GHL is sound and relatively complete under hypotheses, which we formulate independently of any programming language; these are simple conditions which relate the axiom schemata for atomic actions and the axiom schemata, which define the control flow semantics, to the semantics of programs. © 1989 Academic Press, Inc.

1. INTRODUCTION

Lamport (1980) introduced the "Hoare Logic" of concurrent programs. We understand it as a Hoare-style logical system which essentially formalizes Ashcroft's (1975) method for proving invariance properties of parallel processes with shared variables (Ashcroft's method was originally introduced in the style of Floyd (1967)). Moreover, Lamport introduced predicates for reasoning about program locations (replacing Ashcroft's (1975) explicit use of program location counters).

The logic was generalized by Lamport and Schneider (1984) who noticed that the inference rules for non-atomic language constructs (which enable invariance properties of statements to be derived from invariance proper-

0890-5401/89 $3.00

ties of their components) can be viewed as special cases of simple logical meta-rules that apply to all programming constructs. This, in particular, provides a uniform way to compare a variety of invariance proof methods which were designed for different programming constructs.

We propose a formalization of the syntax and semantics of programs (Section 2) which is abstract enough so as to allow a rigorous definition of GHL (Section 3) and its interpretation (Section 4) in a language independent way. We prove that the proof system of GHL is sound (Section 5) and relatively complete (Section 6) under conditions on the semantics of programs and axiom schemata for atomic actions and definition of the control flow semantics which we formulate independently of any particular programming language.

2. Programming Languages

2.1. *Abstract Syntax*

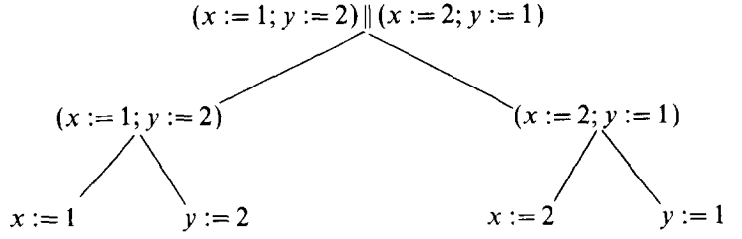
A *programming language* P is a non-empty set of programs. A *program* is made up of declarations (which we ignore but for the fact that for some programming languages they are used to define the set of states of the program and the meaning of the program variables with respect to these states) and an executable program fragment $\Pi \in P \subseteq L$.

A *program fragment* $\pi \in L$ consists of a non-empty finite set $\gamma[\![\pi]\!]$ of smaller program fragments or else is an *atomic action* (in which case $\gamma[\![\pi]\!] = \varnothing$). Hence a program fragment can be understood as a tree with program subfragments as nodes and atomic actions as leaves.

The set of *subfragments* that make up program fragment π is denoted $\gamma^*[\![\pi]\!]$. More precisely, the subfragments of a program fragment π are π itself as well as the subfragments of all the fragments $\pi' \in \gamma[\![\pi]\!]$ of π. In particular, the only subfragment of an atomic action is that atomic action itself.

The set of atomic actions that make up program fragment π is denoted $\alpha[\![\pi]\!]$.

For example, the structure of program $\pi = (x := 1;\ y := 2) \| (x := 2;\ y := 1)$ can be depicted by the tree



so that,

$$\gamma[\![\pi]\!] = \{(x := 1; y := 2), (x := 2; y := 1)\}$$
$$\gamma[\![(x := 1; y := 2)]\!] = \{x := 1, y := 2\}$$
$$\gamma[\![x := 2; y := 1)]\!] = \{x := 2; y := 1\}$$
$$\gamma[\![x := 1]\!] = \gamma[\![y := 2]\!] = \gamma[\![x := 2]\!] = \gamma[\![y := 1]\!] = \varnothing$$

$$\gamma^*[\![\pi]\!] = \{\pi, (x := 1; y := 2), (x := 2; y := 1), \quad x := 1, \quad y := 2, \quad x := 2, y := 1\}$$
$$\alpha[\![\pi]\!] = \{x := 1, y := 2, x := 2, y := 1\}.$$

(The program structure is usually described more precisely using abstract operations (such as ; or ||) between programming constructs. This is not necessary since we are only interested in the set of subfragments of a program fragment.)

More formally, we make the following

HYPOTHESES. (1) *Let L be a non-empty set (of program fragments).*

(2) *Let $P \subseteq L$ be a non-empty set (of programs).*

(3) *Let Υ be a non-empty set (of types or sorts) and V_p be a set (of program variables) such that Υ, V_p, and L are disjoint. Let $\Delta \in (V_p \to \Upsilon)$ be a type assignment (assigning a type $\Delta(v)$ to each program variable $v \in V_p$).*

(4) *Let $\gamma \in (L \to 2^L)$ be such that:*

(a) $\forall \pi \in L$, $\gamma[\![\pi]\!]$ *is finite,*

(b) *There is no infinite chain $\pi_0, \pi_1, ..., \pi_i, ...$ in L such that for all $i \geqslant 0$, $\pi_{i+1} \in \gamma[\![\pi_i]\!]$,*

(c) *If there are chains $\pi_0, ..., \pi_m$ and $\pi'_0, ..., \pi'_n$ in L such that for all $0 \leqslant i < m$, $\pi_{i+1} \in \gamma[\![\pi_i]\!]$, for all $0 \leqslant i < n$, $\pi'_{i+1} \in \gamma[\![\pi'_i]\!]$, $\pi_0 = \pi'_0$, and $\pi_m = \pi'_n$ then $m = n$ and for all $i \in [1, n]$, $\pi_i = \pi'_i$.*

($\gamma[\![\pi]\!]$ *is the set of immediate components of program fragment $\pi \in L$. This set is finite* (a). *No program fragment can be indefinitely decomposed into smaller fragments* (b). *Subfragments of a program fragment are all different* (c).)

DEFINITIONS. (5) $\gamma^* \in (L \to (2^L - \varnothing))$. $\gamma^* = \lambda\pi \cdot \{\pi' \mid \exists \pi_1, ..., \pi_n \in L. (\pi = \pi_1) \wedge (\forall i \in [1, n[\cdot \pi_{i+1} \in \gamma[\![\pi_i]\!]) \wedge (\pi_n = \pi')\}$. ($\gamma^*[\![\pi]\!]$ is the set of all subfragments composing program fragment $\pi \in L$.)

(6) $\alpha \in (L \to (2^L - \varnothing))$, $\alpha = \lambda\pi \cdot \{a \in \gamma^*[\![\pi]\!] \mid \gamma[\![a]\!] = \varnothing\}$. ($\alpha[\![\pi]\!]$ is the set of all atomic actions composing program fragment $\pi \in L$.)

2.2. *Operational Semantics*

Programs can be viewed as defining a *transition relation* on *states* (see (7) and (8) below).

More precisely, execution of a program $\Pi \in P$ is started from some initial state $s_0 \in S[\![\Pi]\!]$ and goes on from one state $s_i \in S[\![\Pi]\!]$ to another state $s_{i+1} \in S[\![\Pi]\!]$ by execution of an atomic action $a \in \alpha[\![\Pi]\!]$. Execution never terminates or stops when reaching a state with no possible successor.

Execution of an atomic action $a \in \alpha[\![\Pi]\!]$ always terminates and is indivisible (so that it cannot pass through any visible intermediate states). Hence a program step $t[\![\Pi]\!](s_i, s_{i+1})$ can be understood as a transition $t[\![a]\!](s_i, s_{i+1})$ between states corresponding to indivisible execution of atomic action "a" of Π, (9).

The states s_i include control information to determine which atomic action (or which actions for non-deterministic programs) can occur next. Following Lamport (1980), we write at$[\![\pi]\!]$ when control resides at an entry point of program fragment $\pi \in \gamma^*[\![\Pi]\!]$; in$[\![\pi]\!]$ when control resides somewhere in π, including at its entry points and after$[\![\pi]\!]$ when control resides at a point immediately following π, but not in$[\![\pi]\!]$. Notice that the starting points of π are considered to be in π but the points at which π may terminate have to be considered to be outside, but not independent, of π (10), (12), (13), (14), (15).

The meaning (hence, in particular, the states) of a program fragment π' cannot be understood independently of its context. More precisely, if program fragment π' is part of a larger fragment π then all states of π for which control resides in or after π' are states relevant to π' (16).

If π is an atomic action, so that there are no places in π at which control can reside except at its starting point (11), the execution of π is possible only when control is at$[\![\pi]\!]$. When execution of an atomic action π terminates, control is after$[\![\pi]\!]$, (17).

When discussing Hoare-style logics, program variables are usually easy to interpret since states are functions assigning to each variable a value from some domain (Apt, 1981). We do not adopt this naïve point of view which, for example, is not powerful enough to describe precisely the execution stack in Pascal. On the contrary, we will understand a program variable as a function which given a program state returns the value of the variable in that state. This value may be undefined and special constants may be used to denote the undefined value of uninitialized variables. Moreover, since states have a control component, the meaning of a program variable may be different at different program places so that scope issues can be taken into account (18), (19).

This intuitive understanding of the operational semantics of programs is made precise by the following:

HYPOTHESES. *For all $\Pi \in P$, an operational semantics is a tuple* $\langle S, t, \text{at}, \text{in}, \text{after}, v \rangle$ *such that for all* $\pi \in \gamma^*[\![\Pi]\!]$, $a \in \alpha[\![\pi]\!]$ *we have*

(7) $S[\![\pi]\!]$ *is a non-empty set (of program states relevant to fragment* π *of program* Π*).*

(8) $t[\![\pi]\!] \in ((S[\![\pi]\!] \times S[\![\pi]\!]) \to \{tt, ff\})$ *(the transition relation for* π*, i.e., a function from pairs of states to truth values (*tt *is true and* ff *is false) describing execution of an elementary step of* π*).*

(9) $t[\![\pi]\!] = \bigvee_{a \in \alpha[\![\pi]\!]} t[\![a]\!]$ *(it is assumed that execution of an elementary step of program fragment* π *corresponds to execution of one of the constituent atomic actions of* π*).*

(10) $\text{at}[\![\pi]\!]$, $\text{in}[\![\pi]\!]$, $\text{after}[\![\pi]\!] \in (S[\![\pi]\!] \to \{tt, ff\})$ *(are functions from states to truth values which respectively characterize those program states for which control resides at, in, and after program fragment* π*).*

(11) $\text{at}[\![a]\!] = \text{in}[\![a]\!]$ *(the only control points inside an atomic action are its entry points).*

(12) $\text{at}[\![\pi]\!] \Rightarrow \text{in}[\![\pi]\!]$ *(the starting points of* π *are considered to be inside* π*).*

(13) *if* $\pi' \in \gamma^*[\![\pi]\!]$ *then* $\text{in}[\![\pi']\!] \Rightarrow \text{in}[\![\pi]\!]$ *(if control resides somewhere in a subfragment* π' *of program fragment* π *then control also resides in* π*).*

(14) $\text{after}\ [\![\pi]\!] \Rightarrow \neg \text{in}[\![\pi]\!]$ *(the points at which program fragment* π *terminates are considered to be outside* π*).*

(15) *if* $\pi' \in \gamma^*[\![\pi]\!]$ *then* $\text{after}[\![\pi']\!] \Rightarrow (\text{in}[\![\pi]\!] \vee \text{after}[\![\pi]\!])$ *(if control resides after a subfragment* π' *of program fragment* π *then control resides in* π *or after* π*).*

(16) $\forall \pi' \in \gamma^*[\![\pi]\!].\ S[\![\pi']\!] = \{s \in S[\![\pi]\!] \mid \text{in}[\![\pi']\!](s) \vee \text{after}[\![\pi']\!](s)\}$ *(it is assumed that the program states relevant to subfragment* π' *of program fragment* π *are those states of* π *for which control resides in or after* π'*).*

(17) $\forall s, s' \in S[\![a]\!].\ t[\![a]\!](s, s') \Rightarrow (\text{at}[\![a]\!](s) \wedge \text{after}[\![a]\!](s'))$ *(it is possible to execute an atomic action "*a*" only if control resides at an entry point of "*a*." Moreover, after execution of "*a*," control must reside after "*a*," that is, at one of the exit points of "*a*").*

(18) *For every sort* $\mathbf{s}$ *in* Υ, $v(\mathbf{s})$ *is a non-empty set (of values of program variables of type* $\mathbf{s}$*).*

(19) *For every program variable* $x \in V_p$ *of type* $\Delta(x)$, $v[\![x]\!]$ *is a function* $S[\![\Pi]\!] \to v(\Delta(x))$.

(The meaning of variable x *in state* s *is* $v[\![x]\!](s)$*).*

From these hypotheses we derive the following theorem which will be very useful later on:

THEOREM. (20) $\forall s,\ s' \in S[\![\pi]\!].\ t[\![\pi]\!](s, s') \Rightarrow [\mathrm{in}[\![\pi]\!](s) \wedge (\mathrm{in}[\![\pi]\!](s') \vee \mathrm{after}[\![\pi]\!](s'))]$. (Execution of an elementary program step inside π is only possible when control resides in π. Afterwards, control remains in π or resides at a point immediately following π.)

Proof. $t[\![\pi]\!](s, s') \Rightarrow^{(9)} (\exists a \in \alpha[\![\pi]\!],\ t[\![a]\!](s, s')) \Rightarrow^{(17)} (\exists a \in \alpha[\![\pi]\!],\ \mathrm{at}[\![a]\!](s) \wedge \mathrm{after}[\![a]\!](s')) \Rightarrow^{(11)} (\exists a \in \alpha[\![\pi]\!],\ \mathrm{in}[\![a]\!](s) \wedge \mathrm{after}[\![a]\!](s')) \Rightarrow^{(6)} (\exists a \in \gamma^*[\![\pi]\!], \mathrm{in}[\![a]\!](s) \wedge \mathrm{after}[\![a]\!](s')) \Rightarrow^{(13),(15)} (\mathrm{in}[\![\pi]\!](s) \wedge (\mathrm{in}[\![\pi]\!](s') \vee \mathrm{after}[\![\pi]\!](s')))$. ∎

3. THE LOGIC GHL

3.1. *The Language of GHL*

As usual for Hoare-style program logics (Apt, 1981, 1982) the formulas of GHL include assertions (or predicates on the values of the program variables and program locations) and asserted programs of the form $\{P\}\ \pi\{Q\}$ having the following interpretation (which differs from Hoare's $P\{\pi\}\ Q$): if execution is begun anywhere in π with the assertion P true, then executing π will leave P true while control resides inside π and will make Q true if and when S terminates.

3.1.1. *Assertions*

Let A be a (maybe infinitary) many-sorted first-order logic, (Feferman, 1974).

More precisely, we consider a collection of symbols falling into the following disjoint classes:

(21) A non-empty set, the elements of which are called *sorts*, which contains $\mathbf{b}$ (the sort of truth values), Υ (the sorts of program variables).

(22) A set of finitary sorted *relation symbols* (each relation symbol r is equipped with some sort $r: \mathbf{s}_1 \times \cdots \times \mathbf{s}_{\#r} \to \mathbf{b}$). In particular, *true*, *false*, and for all $\Pi \in P$, $\pi \in \gamma^*[\![\pi]\!]$, $at[\![\pi]\!]$, $in[\![\pi]\!]$ and $after[\![\pi]\!]$ are 0-ary symbols of sort $\to \mathbf{b}$.

(23) A set of finitary sorted *function symbols* (each function symbol f is equipped with some sort $f: \mathbf{s}_1 \times \cdots \times \mathbf{s}_{\#f} \to \mathbf{s}$), 0-ary function symbols are called *individual constants*.

(24) A set V^f of free variables which is partitioned into a set V_l^f of logical free variables (which may appear in proofs but not in programs) and a set $V_p^f \subseteq V_p$ of program free variables (which may appear both in proofs and programs). The sort of free variable $x \in V^f$ is $\Delta(x)$.

(25) A set V^b of bound variables partitioned into V_l^b and $V_p^b \subseteq V_p$. The sort of bound variable $x \in V^b$ is $\Delta(x)$.

(26) The symbols $\equiv$ for identity, $\vee$ (maybe infinitary) for disjunctions, $\wedge$ (maybe infinitary) for conjonction, $\neg$ for negation, $\Rightarrow$ for implication, $\exists$ existential quantifier, $\forall$ universal quantifier.

In order to define the class of assertions we introduce as usual:

(27) The class of *terms of sort* $\mathbf{s}$ is the least class containing the free variables and individual constants of sort $\mathbf{s}$ and the expressions of the form $f(t_1, ..., t_n)$, where $n = \#f > 0$, $f: \mathbf{s}_1 \times \cdots \times \mathbf{s}_n \to \mathbf{s}$, and the t_i are terms of sort $\mathbf{s}_i$.

(28) The *atomic assertions* are expressions of the form $r(t_1, ..., t_n)$, where the t_i are terms of sort $\mathbf{s}_i$, $r: \mathbf{s}_1 \times \cdots \times \mathbf{s}_n \to \mathbf{b}$, and $n = \#r$, or of the form $t_1 \equiv t_2$, where t_1 and t_2 have the same sort.

We take for granted the notion of *substitution*: $\phi(w/x)$ denotes the result of substituting w for x at each occurrence of the free variable x in ϕ.

(29) The class of *assertions* is the least class A such that:

- A contains all atomic assertions,
- A contains $\neg P$, $P \Rightarrow Q$, $P \equiv Q$, $\vee \theta$, $\wedge \theta$ whenever $P, Q \in A$ and $\theta \subset A$ is a set with altogether finitely many free variables occurring in the formulas in θ,
- A contains $\exists w \cdot P(w/x)$ and $\forall w \cdot P(w/x)$ whenever x is a variable actually occurring as a free variable in $P \in A$, w is a bound variable not occurring in P and x and w have the same sort $\Delta(x) = \Delta(w)$.

In the following we will usually suppress our distinction between free and bound variables so that we refer to $\exists x.\phi$, x being a free variable in ϕ, meaning $\exists w.\phi(w/x)$. (However, the distinction between logical and program variables is essential since they have different interpretations.)

3.1.2. *Asserted Programs*

Formulas of GHL are either assertions of A or are *asserted programs* of the form $\{P\}\pi\{Q\}$, where P, Q are assertions and π is a program fragment:

(30) $F = \{\{P\}\pi\{Q\} \mid P, Q \in A, \exists \Pi \in P.\ \pi \in \gamma^*[\![\Pi]\!]\}$.

3.2. *The Formal System of GHL*

The basic proof system of GHL consists of a formal system τ concerning assertions of A (31) and a formal system H concerning asserted programs of F.

τ must contain axiom schemata specifying the control flow semantics of program fragments. In Lamport (1980), examples of such axiom schemata

are given which specify the relation between the predicates $loc[\![\pi]\!]$ (where $loc[\![\pi]\!]$ denotes any one of the predicates $at[\![\pi]\!]$, $in[\![\pi]\!]$, and $after[\![\pi]\!]$) for the entire program fragment π and predicates $loc[\![\pi']\!]$ for its component subfragments $\pi' \in \gamma^*[\![\pi]\!]$. An extension to language independent general axiom schemata would be very useful, but we have found no sound and complete one. (For example, one can attempt to reduce control predicates to more elementary control predicates upon control locations of atomic actions. Then a general axiom schema of the form $in[\![\pi]\!] = \bigvee_{a \in \alpha[\![\pi]\!]} at[\![\pi]\!]$ would be correct for sequential programs but does not work for concurrent programs with multiple program counters). Therefore we have to assume that GHL includes for each non-atomic program fragment, a specification of its *loc* predicates, which serve to define its control flow semantics (32), (33), (34).

In the same way, it is hopeless to look for a language independent specification of the effect of atomic actions in H. One can only make very general hypotheses upon the form of such specifications, (37).

Apart from these language dependent axiom schemata, GHL can be defined by simple language independent meta-rules (35), (36), (38),..., (43).

3.2.1. *The Formal System τ for Assertions of A*

(31) τ is an Hilbert-style formal system for assertions of A augmented with the following axiom schemata (specifying the control flow semantics of program fragments):

(32) For all $\Pi \in P$, $a \in \alpha[\![\Pi]\!]$,

$$at[\![a]\!] \equiv in[\![a]\!].$$

For all $\Pi \in P$, $\pi \in (\gamma^*[\![\Pi]\!] - \alpha[\![\Pi]\!])$,

$$at[\![\pi]\!] \equiv \mathrm{AT}[\![\pi]\!](loc[\![\pi']\!] \mid \pi' \in \gamma[\![\pi]\!] \vee \pi \in \gamma[\![\pi']\!]) \wedge in[\![\pi]\!].$$

(33) For all $\Pi \in P$, $a \in \alpha[\![\Pi]\!]$,

$$in[\![a]\!] \Rightarrow \bigwedge_{a \in \gamma^*[\![\pi]\!]} in[\![\pi]\!].$$

For all $\Pi \in P$, $\pi \in (\gamma^*[\![\Pi]\!] - \alpha[\![\Pi]\!])$,

$$in[\![\pi]\!] \equiv \mathrm{IN}[\![\pi]\!](loc[\![\pi']\!] \mid \pi' \in \gamma[\![\pi]\!] \vee \pi \in \gamma[\![\pi']\!]) \wedge \bigwedge_{\pi \in \gamma^*[\![\pi']\!]} in[\![\pi']\!].$$

(34) For all $\Pi \in P$, $a \in \alpha[\![\Pi]\!]$,

$$after[\![a]\!] \Rightarrow \neg in[\![a]\!] \wedge \bigwedge_{a \in \gamma^*[\![\pi]\!]} (in[\![\pi]\!] \vee after[\![\pi]\!]).$$

For all $\Pi \in P$, $\pi \in (\gamma^*[\![\Pi]\!] - \alpha[\![\Pi]\!])$,

$$after[\![\pi]\!] \equiv \mathrm{AFTER}[\![\pi]\!](loc[\![\pi']\!] \mid \pi' \in \gamma[\![\pi]\!] \vee \pi \in \gamma[\![\pi']\!]) \wedge \neg in[\![\pi]\!]$$

$$\wedge \bigwedge_{\pi \in \gamma^*[\![\pi']\!]} (in[\![\pi']\!] \vee after[\![\pi']\!]).$$

(The control predicates of a program fragment π (i.e., $at[\![\pi]\!]$, $in[\![\pi]\!]$, $after[\![\pi]\!]$) are defined by an assertion of A ($\mathrm{AT}[\![\pi]\!]$, $\mathrm{IN}[\![\pi]\!]$, $\mathrm{AFTER}[\![\pi]\!]$) which only depends upon the control predicates of the immediate components π' of π or upon the control predicates of the program fragments π' which have π as immediate component (i.e., $\{loc[\![\pi']\!] \mid \pi' \in \gamma[\![\pi]\!] \vee \pi \in \gamma[\![\pi']\!]\}$).)

3.2.2. *The Formal System H for Asserted Programs of F*

3.2.2.1. *Axiom schemata of H. For all* $\Pi \in P$, $\pi \in \gamma^*[\![\Pi]\!]$, $a \in \alpha[\![\pi]\!]$, P, Q, R, $I \in A$, $n \geqslant 1$, $P_1, \ldots, P_n, Q_1, \ldots, Q_n \in A$,

(35) $\{in[\![\pi]\!]\}\ \pi\{true\}$

(36) $\{true\}\ \pi\{after[\![\pi]\!]\}$

(37a) $\{\mathrm{PRE}[\![a]\!](Q)\}\ a\{Q\}$, where $\mathrm{PRE}[\![a]\!] \in (A \to A)$. ($\mathrm{PRE}[\![a]\!](Q)$ is a (preferably the weakest) pre-condition on those values of the program variables corresponding to a state for which control resides at an entry point of atomic action "a" such that in the successor state after activation and termination of "a," the values of the program variables corresponding to that successor state satisfy Q.)

(37b) $\{P\}a\{\mathrm{POST}[\![a]\!](P)\}$, where $\mathrm{POST}[\![a]\!] \in (A \to A)$. ($\mathrm{POST}[\![a]\!](P)$ is (preferably the strongest) post-condition corresponding to pre-condition P.)

3.2.2.2. *Rules of inference of H.*

(38) (Decomposition principle)

$$\frac{\{I\}\ \pi'\{I\} \text{ for all } \pi' \in \gamma[\![\pi]\!]}{\{I\}\ \pi\{I\}}.$$

(39) (Locality rule)

$$\frac{\{in[\![\pi]\!] \wedge P\}\ \pi\{after[\![\pi]\!] \wedge Q\}}{\{P\}\pi\{Q\}}.$$

(40) (Conjunction rule)

$$\frac{\{P_1\}\ \pi\{Q_1\}, \ldots, \{P_n\}\ \pi\{Q_n\}}{\{\bigwedge_{i=1}^n P_i\}\ \pi\{\bigwedge_{i=1}^n Q_i\}}, \qquad n \geqslant 1.$$

(41) (Right consequence rule)

$$\frac{\{P\}\ \pi\{Q\},\ Q \Rightarrow R}{\{P\}\ \pi\{R\}}.$$

(42) (Left identity rule)

$$\frac{R \equiv P,\ \{P\}\ \pi\{Q\}}{\{R\}\ \pi\{Q\}}.$$

(43) (Left consequence rule)

$$\frac{R \Rightarrow P,\ \{P\}\ a\{Q\}}{\{R\}\ a\{Q\}}.$$

(As noticed by Lamport (1980) and contrary to Hoare's logic, this derivation does not hold for non-atomic actions. For a counterexample we have $\{x \geqslant 0\}\, x := x+1; x := x+2\ \{true\}$ and $((x=0) \Rightarrow (x \geqslant 0))$, but in GHL we do not have $\{x=0\}\ x := x+1; x := x+2\ \{true\}$, since assuming $x=0$ at $x := x+1$ we have $x \neq 0$ after $x := x+1$.)

3.3. *Proofs in GHL*

(44) A proof of formula Φ from a set A of formulas in a formal system Ξ is a finite sequence of formulas $\Psi_1, ..., \Psi_n$ with $\Psi_n = \Phi$ each of which is either an axiom of Ξ, a member of A or else follows from earlier Ψ_i by one of the rules of inference of Ξ. If there is a proof of Φ from A in Ξ we say that Φ is provable from A in Ξ and write

$$A \vdash_{\Xi} \Phi.$$

In particular, when A is empty we write

$$\vdash_{\Xi} \Phi.$$

3.4. *Examples of Proofs in GHL*

We give a number of very simple (meta-) proofs which will be useful later. We leave implicit the fact that they use tautologies, equality axioms, modus ponens,... of τ:

(45) (a) $\{P\}\ \pi\{Q\} \vdash_{\tau, H} \{in[\![\pi]\!] \wedge P\}\ \pi\{after[\![\pi]\!] \wedge Q\}$.

Proof.

(b) $\{in[\![\pi]\!]\}\ \pi\{true\}$, by (35);

(c) $\{true\}\ \pi\{after[\![\pi]\!]\}$, by (36);

(d) $\{P \wedge in[\![\pi]\!] \wedge true\}\ \pi\{Q \wedge true \wedge after[\![\pi]\!]\}$, by (a), (b), (c), (40);

(e) $P \wedge in[\![\pi]\!] \wedge true \equiv in[\![\pi]\!] \wedge P$, by definition of τ;

(f) $\{in[\![\pi]\!] \wedge P\}\ \pi\{Q \wedge true \wedge after[\![\pi]\!]\}$, by (e), (d), (42);

(g) $Q \wedge true \wedge after[\![\pi]\!] \Rightarrow after[\![\pi]\!] \wedge Q$, by τ;

(h) $\{in[\![\pi]\!] \wedge P\}\ \pi\{after[\![\pi]\!] \wedge Q\}$, by (f), (g), (41). ∎

(46) (a) $\{P\}\ \pi\{Q\} \vdash_{\tau, H} \{in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q\}\ \pi\{in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q\}$.

Proof.

(b) $\{in[\![\pi]\!] \wedge P\}\ \pi\{after[\![\pi]\!] \wedge Q\}$, by (a), (45);

(c) $after[\![\pi]\!] \Rightarrow \neg in[\![\pi]\!]$, by (34), τ;

(d) $in[\![\pi]\!] \Rightarrow \neg after[\![\pi]\!]$, by (c), τ;

(e) $in[\![\pi]\!] \wedge P \equiv in[\![\pi]\!] \wedge (in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q)$, by (d), τ;

(f) $\{in[\![\pi]\!] \wedge (in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q)\}\ \pi\ \{after[\![\pi]\!] \wedge Q\}$, by (e), (b), (42);

(g) $after[\![\pi]\!] \wedge Q \Rightarrow after[\![\pi]\!] \wedge (in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q)$, by (c), τ;

(h) $\{in[\![\pi]\!] \wedge (in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q)\}\pi\{after[\![\pi]\!] \wedge (in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q)\}$, by (g), (f), (41);

(i) $\{in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q\}\ \pi\{in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q\}$, by (h), (39). ∎

(47) (a) $\{in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q\}\ \pi\{in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q\} \vdash_{\tau, H} \{P\}\ \pi\{Q\}$.

Proof.

(b) $\{in[\![\pi]\!] \wedge (in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q)\}\pi\ \{after[\![\pi]\!] \wedge (in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q)\}$, by (a), (45);

(c) $in[\![\pi]\!] \wedge (in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q) \equiv in[\![\pi]\!] \wedge P$, by (34), τ;

(d) $\{in[\![\pi]\!] \wedge P\}\ \pi\{after[\![\pi]\!] \wedge (in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q)\}$, by (c), (b), (42);

(e) $(after[\![\pi]\!] \wedge (in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q)) \Rightarrow (after[\![\pi]\!] \wedge Q)$, by (34), τ;

(f) $\{in[\![\pi]\!] \wedge P\}\ \pi\ \{after[\![\pi]\!] \wedge Q\}$, by (d), (e), (41);

(g) $\{P\}\ \pi\{Q\}$, by (39). ∎

(48) If $I \in A$, $n \geqslant 1$, $\pi_1, ..., \pi_n \in \gamma^*[\![\pi]\!]$, $\alpha[\![\pi]\!] \subseteq \bigcup_{i=1}^{n} \alpha[\![\pi_i]\!]$ then

$$\{I\}\,\pi_1\{I\},..., \{I\}\,\pi_n\{I\} \vdash_H \{I\}\,\pi\{I\}.$$

(Contrary to Lamport and Schneider's (1984) decomposition principle, we do not only require that the atomic actions of π are just the atomic actions of the π_i together. We also require that the π_i be subfragments of π. Otherwise the inference could be incorrect as shown by the following counterexample, where I is $a=0$, π_1 is $y:=1$, π_2 is $y:=0$; $a:=y$ and π is $y:=0$; $y:=1$; $a:=y$ so that $\alpha[\![\pi]\!]=\alpha[\![\pi_1]\!]\cup\alpha[\![\pi_2]\!]$, $\{I\}\,\pi_1\{I\}$ and $\{I\}\,\pi_2\{I\}$ are true but $\{I\}\,\pi\{I\}$ is not).

Proof. Let us inductively build a (syntactic) tree of π as follows:

- Initially, π is the root of the tree.
- At each step, add to all leaves π' of the tree which are not among $\pi_1, ..., \pi_n$ their immediate descendants $\gamma[\![\pi']\!]$.

By hypotheses (4a), (4b), and Koening's lemma the resulting tree is finite.

We now prove that all leaves in this tree are among $\pi_1, ..., \pi_n$ so as to be able to later prove (48) by induction using a traversal of the tree in postfixed order.

To prove that all leaves in the tree are among $\pi_1, ..., \pi_n$ we proceed by reductio ad absurdum. Assuming that π' is a leave not in $\pi_1, ..., \pi_n$ we construct two different chains $\pi, \pi'_1, ..., \pi'_{p-1}, \pi'$ and $\pi, \pi''_1, ..., \pi''_{m-1}, \pi'$ in contradiction with hypothesis (4c).

To define the first chain, observe that by the construction process $\pi' \in \gamma^*[\![\pi]\!]$ and $\gamma[\![\pi']\!]=\varnothing$ so that by (6), $\pi' \in \alpha[\![\pi]\!]$. Since $\alpha[\![\pi]\!] \subseteq \bigcup_{i=1}^{n} \alpha[\![\pi_i]\!]$, we have $\pi' \in \alpha[\![\pi_i]\!]$ for some $i \in [1, n]$. Since $\pi' \in \alpha[\![\pi_i]\!]$ and $\pi_i \in \gamma^*[\![\pi]\!]$, there is by definitions (5) and (6) a chain of the form $\pi=\pi'_0, ..., \pi'_k=\pi_i, ..., \pi'_p=\pi'$ such that $k<p$ (since otherwise $\pi_i=\pi'$) and such that $\pi'_j \in \gamma[\![\pi'_{j+1}]\!]$ for $j=0,..., p-1$. To define the second chain, observe that by the tree construction process there is a chain of the form $\pi=\pi''_0, ..., \pi''_m=\pi'$ such that $\pi_i \neq \pi''_j$ for $j=0, ..., m$ (since otherwise by the construction process $\pi''_m=\pi_i$ and again $\pi_i=\pi'$) and $\pi''_j \in \gamma[\![\pi''_{j+1}]\!]$ for $j=0, ..., m-1$. The fact that the two chains $\pi, \pi'_1, ..., \pi'_{p-1}, \pi'$ and $\pi, \pi''_1, ..., \pi''_{m-1}, \pi'$ are different (since π_i belongs to the first and does not appear in the second) is contrary to hypothesis (4c) so that by reductio ad absurdum we have proved that all leaves in the tree are in $\pi_1, ..., \pi_n$.

We can now built the proof of $\{I\}\,\pi\,\{I\}$ as follows: Initially we have $\{I\}\,\pi_i\{I\}$ for $i=1, ..., m$ and treated all leaves of the tree. At each step we treat a node π' such that its sons $\pi'' \in \gamma[\![\pi']\!]$ have already been treated. Unless we are done ($\pi'=\pi$) this choice is always possible because we have a finite tree and so each node has a finite number of sons. Hence from $\{I\}\,\pi''\{I\}$ for all $\pi'' \in \gamma[\![\pi']\!]$ and (38) we derive $\{I\}\,\pi'\{I\}$. This algorithm terminates because the tree is finite so that at the root we have proved $\{I\}\,\pi\{I\}$. ∎

Let us give two additional proofs in GHL, about sequential and parallel composition, which show how to extend GHL to particular programming constructs:

(49) *Sequential composition.* Assume $\Pi \in P$, $\gamma[\![\pi_1; \pi_2]\!] \in \gamma^*[\![\Pi]\!]$ and $\gamma[\![\pi_1; \pi_2]\!] = \{\pi_1, \pi_2\}$ then

(a) $at[\![\pi_1; \pi_2]\!] \equiv at[\![\pi_1]\!]$

(b) $after[\![\pi_1]\!] \equiv at[\![\pi_2]\!]$

(c) $in[\![\pi_1; \pi_2]\!] \equiv (in[\![\pi_1]\!] \vee in[\![\pi_2]\!]) \wedge \neg(in[\![\pi_1]\!] \wedge in[\![\pi_2]\!])$

(d) $after[\![\pi_1; \pi_2]\!] \equiv after[\![\pi_2]\!]$

(e) $\{P\}\ \pi_1\{Q\}$

(f) $\{R\}\ \pi_2\{U\}$

(g) $Q \wedge at[\![\pi_2]\!] \Rightarrow R \vdash_{\tau, H} \{in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R\}\ \pi_1; \pi_2\{U\}$.

Proof.

(h) $\{in[\![\pi_1]\!] \wedge P\}\ \pi_1\{after[\![\pi_1]\!] \wedge Q\}$, by (e), (45);

(i_1) $in[\![\pi_1]\!] \Rightarrow in[\![\pi_1; \pi_2]\!]$, by (33) and $\pi_1 \in \gamma[\![\pi_1; \pi_2]\!]$;

(i_2) $in[\![\pi_1]\!] \Rightarrow \neg in[\![\pi_2]\!]$, by ($i_1$), (c), τ;

(i_3) $in[\![\pi_1; \pi_2]\!] \Rightarrow \neg after[\![\pi_1; \pi_2]\!]$, by (34), τ;

(i_4) $\neg after[\![\pi_1; \pi_2]\!] \Rightarrow \neg after[\![\pi_2]\!]$, by (d), τ;

(i_5) $in[\![\pi_1]\!] \Rightarrow \neg after[\![\pi_2]\!]$, by ($i_1$,), ($i_3$), ($i_4$), τ;

(i) $in[\![\pi_1]\!] \wedge P \equiv in[\![\pi_1]\!] \wedge (in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R \wedge after[\![\pi_2]\!] \Rightarrow U)$, by ($i_2$), ($i_5$), τ;

(j_1) $after[\![\pi_1]\!] \Rightarrow \neg in[\![\pi_1]\!]$, by (34), τ;

(j_2) $after[\![\pi_1]\!] \wedge Q \equiv at[\![\pi_2]\!] \wedge Q$, by (b), τ;

(j_3) $at[\![\pi_2]\!] \wedge Q \Rightarrow at[\![\pi_2]\!] \wedge R$, by (g), τ;

(j_4) $at[\![\pi_2]\!] \wedge R \Rightarrow in[\![\pi_2]\!] \wedge R$, by (32), τ;

(j_5) $in[\![\pi_2]\!] \wedge R \Rightarrow (in[\![\pi_2]\!] \Rightarrow R)$, by τ;

(j_6) $after[\![\pi_1]\!] \wedge Q \Rightarrow (in[\![\pi_2]\!] \Rightarrow R)$, by ($j_2$), ..., ($j_5$), τ;

(j_7) $after[\![\pi_1]\!] \Rightarrow at[\![\pi_2]\!]$, by (b), τ;

(j_8) $at[\![\pi_2]\!] \Rightarrow in[\![\pi_2]\!]$ by (32), τ;

(j_9) $in[\![\pi_2]\!] \Rightarrow \neg after[\![\pi_2]\!]$, by (34), τ;

(j_{10}) $after[\![\pi_1]\!] \Rightarrow \neg after[\![\pi_2]\!]$, by ($j_7$), ($j_8$), ($j_9$), τ;

(j) $after[\![\pi_1]\!] \wedge Q \Rightarrow after[\![\pi_1]\!] \wedge (in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R \wedge after[\![\pi_2]\!] \Rightarrow U)$, by ($j_1$), ($j_6$), ($j_{10}$), τ;

(k) $\{in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow \mathrm{R} \wedge after[\![\pi_2]\!] \Rightarrow U\}\ \pi_1\{in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R \wedge after[\![\pi_2]\!] \Rightarrow U\}$, by (h), (i), (42), (j), (41), (39);

(l) $\{in[\![\pi_2]\!] \wedge R\}\ \pi_2\{after[\![\pi_2]\!] \wedge U\}$, by (f), (45);

(m_1) $in[\![\pi_2]\!] \Rightarrow \neg in[\![\pi_1]\!]$, by ($i_2$), τ;

(m_2) $in[\![\pi_2]\!] \Rightarrow \neg after[\![\pi_2]\!]$, by (34), τ;

(m) $in[\![\pi_2]\!] \wedge R \equiv in[\![\pi_2]\!] \wedge (in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R \wedge after[\![\pi_2]\!] \Rightarrow U)$, by ($m_1$), ($m_2$), τ;

(n_1) $after[\![\pi_2]\!] \Rightarrow \neg in[\![\pi_1]\!]$, by ($i_5$), τ;

(n_2) $after[\![\pi_2]\!] \Rightarrow \neg in[\![\pi_2]\!]$, by (34), τ;

(n) $after[\![\pi_2]\!] \wedge U \Rightarrow after[\![\pi_2]\!] \wedge (in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R \wedge after[\![\pi_2]\!] \Rightarrow U)$, by ($n_1$), ($n_2$), τ;

(o) $\{in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R \wedge after[\![\pi_2]\!] \Rightarrow U\}\ \pi_2\ \{in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R \wedge after[\![\pi_2]\!] \Rightarrow U\}$, by (l), (m), (42), (n), (41), (39);

(p) $\{in[\![\pi_1; \pi_2]\!] \wedge (in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R \wedge after[\![\pi_2]\!] \Rightarrow U)\}\ \pi_1; \pi_2\ \{after[\![\pi_1; \pi_2]\!] \wedge (in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R \wedge after[\![\pi_2]\!] \Rightarrow U)\}$, by (k), (l), (38), (45);

(q_1) $in[\![\pi_1; \pi_2]\!] \Rightarrow \neg after[\![\pi_2]\!]$, by ($i_3$), ($i_4$), τ;

(q_2) $in[\![\pi_1; \pi_2]\!] \wedge (in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R \wedge after[\![\pi_2]\!] \Rightarrow U) \equiv in[\![\pi_1; \pi_2]\!] \wedge (in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R)$, by ($q_1$), τ;

(q_3) $after[\![\pi_1; \pi_2]\!] \wedge (in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R \wedge after[\![\pi_2]\!] \Rightarrow U) \Rightarrow after[\![\pi_1; \pi_2]\!] \wedge U$, by (d), τ;

(q) $\{in[\![\pi_1]\!] \Rightarrow P \wedge in[\![\pi_2]\!] \Rightarrow R\}\ \pi_1; \pi_2\{U\}$, by (p), ($q_2$), (42), ($q_3$), (41), (39). ∎

(50) *Parallel composition.* Assume $\Pi \in P$, $\pi_1 \| \pi_2 \in \gamma^*[\![\Pi]\!]$ and $\gamma[\![\pi_1 \| \pi_2]\!] = \{\pi_1, \pi_2\}$ then

(a) $at[\![\pi_1 \| \pi_2]\!] = at[\![\pi_1]\!] \wedge at[\![\pi_2]\!]$

(b) $in[\![\pi_1 \| \pi_2]\!] = (in[\![\pi_1]\!] \vee after[\![\pi_1]\!]) \wedge (in[\![\pi_2]\!] \vee after[\![\pi_2]\!]) \wedge \neg after[\![\pi_1 \| \pi_2]\!]$

(c) $after[\![\pi_1 \| \pi_2]\!] = after[\![\pi_1]\!] \wedge after[\![\pi_2]\!]$

(d) $\{I_1\}\ \pi_1\{I_1\}$

(e) $I_1 \wedge after[\![\pi_1]\!] \Rightarrow Q$

(f) $\{I_2\}\ \pi_2\{I_2\}$

(g) $I_2 \wedge after[\![\pi_2]\!] \Rightarrow Q$

(h) $\{(in[\![\pi_2]\!] \vee after[\![\pi_2]\!]) \wedge I_2\}\ \pi_1\{I_2\}$

(i) $\{(in[\![\pi_1]\!] \vee after[\![\pi_1]\!]) \wedge I_1\}\ \pi_2\{I_1\} \vdash_{\tau, H} \{I_1 \wedge I_2\}\ \pi_1 \| \pi_2\{Q\}$.

Proof.

(j_1) $\{in[\![\pi_1]\!] \wedge I_1\}\ \pi_1\{after[\![\pi_1]\!] \wedge I_1\}$, by (d), (45);

(j_2) $in[\![\pi_1]\!] \wedge I_1 \equiv in[\![\pi_1]\!] \wedge (in[\![\pi_1]\!] \vee after[\![\pi_1]\!]) \wedge I_1$, by τ;

(j_3) $\{in[\![\pi_1]\!] \wedge (in[\![\pi_1]\!] \vee after[\![\pi_1]\!]) \wedge I_1\}\ \pi_1\{after[\![\pi_1]\!] \wedge I_1\}$, by ($j_1$), ($j_2$), (42);

(j_4) $\{(in[\![\pi_1]\!] \vee after[\![\pi_1]\!]) \wedge I_1\}\ \pi_1\{I_1\}$, by ($j_3$), (39);

(j) $\{(in[\![\pi_1]\!] \vee after[\![\pi_1]\!]) \wedge I_1 \wedge (in[\![\pi_2]\!] \vee after[\![\pi_2]\!]) \wedge I_2\}$ $\pi_1\{I_1 \wedge I_2\}$, by (j_4), (h), (40);

(k) $\{(in[\![\pi_1]\!] \vee after[\![\pi_1]\!]) \wedge I_1 \wedge (in[\![\pi_2]\!] \vee after[\![\pi_2]\!]) \wedge I_2\}$ $\pi_2\{I_1 \wedge I_2\}$, by (f), (45), τ, (42), (39), (i), (40);

(l) $\{in[\![\pi_1 \| \pi_2]\!] \wedge (in[\![\pi_1]\!] \vee after[\![\pi_1]\!]) \wedge I_1 \wedge (in[\![\pi_2]\!] \vee after[\![\pi_2]\!]) \wedge I_2\}\ \pi_1 \| \pi_2\{after[\![\pi_1 \| \pi_2]\!] \wedge I_1 \wedge I_2\}$, by (j), (k), (38);

(m) $in[\![\pi_1 \| \pi_2]\!] \wedge (in[\![\pi_1]\!] \vee after[\![\pi_1]\!]) \wedge I_1 \wedge (in[\![\pi_2]\!] \vee after[\![\pi_2]\!]) \wedge I_2 \equiv in[\![\pi_1 \| \pi_2]\!] \wedge I_1 \wedge I_2$, by (b), τ;

(n) $\{in[\![\pi_1 \| \pi_2]\!] \wedge I_1 \wedge I_2\}\ \pi_1 \| \pi_2\ \{after[\![\pi_1 \| \pi_2]\!] \wedge I_1 \wedge I_2\}$, by (m), (l), (42);

(o) $after[\![\pi_1 \| \pi_2]\!] \wedge I_1 \wedge I_2 \Rightarrow after[\![\pi_1 \| \pi_2]\!] \wedge Q$, by (c), (e), (g), τ;

(p) $\{I_1 \wedge I_2\}\ \pi_1 \| \pi_2\ \{Q\}$, by (o), (41), (39). ∎

The above processes can communicate only using shared variables. In order to consider CSP-like programs, we must also consider sending actions $\vec{L} \subseteq L$ (such as $P_i!\,e$ in CSP) and receiving actions $\overleftarrow{L} \subseteq L$. Some sending and receiving actions match to perform channel communications $\vec{L} \subseteq \vec{L} \times \overleftarrow{L} \subseteq L$.

Since sending and receiving actions of a parallel program $\pi_1 \| \pi_2$ cannot be executed separately they are not considered as subfragments of $\pi_1 \| \pi_2$. To do so they have to be grouped into matching pairs performing channel communications. Therefore,

$$\gamma^*[\![\pi_1 \| \pi_2]\!] = (\gamma^*[\![\pi_1]\!] \cup \gamma^*[\![\pi_2]\!]) - \vec{L} - \overleftarrow{L} \cup \vec{L} \cap [(\gamma^*[\![\pi_1]\!] \cup \gamma^*[\![\pi_2]\!]) \cap \vec{L} \times (\gamma^*[\![\pi_1]\!] \cup \gamma^*[\![\pi_2]\!]) \cap \overleftarrow{L}].$$

Applying the decomposition principle we can prove $\{I\}\ \pi_1 \| \pi_2\{I\}$ by first considering the concurrent program without any communication over channels (or equivalently consider operations of $\vec{L} \cup \overleftarrow{L}$ as "halts"):

$$\forall \pi' \in (\gamma^*[\![\pi_1]\!] \cup \gamma^*[\![\pi_2]\!]) - \vec{L} - \overleftarrow{L}. \qquad \{I\}\ \pi'\{I\}$$

and then, considering channel communications in a cooperation proof,

$$\forall \pi' \in \vec{L} \cap [(\gamma^*[\![\pi_1]\!] \cup \gamma^*[\![\pi_2]\!]) \cap \vec{L} \times (\gamma^*[\![\pi_1]\!] \cup \gamma^*[\![\pi_2]\!]) \cap \vec{L}]. \qquad \{I\}\ \pi'\ \{I\}.$$

4. Interpretation of GHL

So far the formulas of GHL are simply finite strings of symbols. We now assign a meaning to formulas of GHL. This is done by defining a *satisfaction relation* $\models_I \phi$ between *interpretations* I of logical symbols and programs on the one hand and formulas ϕ of GHL on the other.

An interpretation I of GHL for a program $\Pi \in P$ is a pair $\langle M, S \rangle$ where:

(51) M is a set-theoretic structure of A (which assigns an appropriate meaning to the logical symbols of A).

(52) $S = \langle S, t, \text{at}, \text{in}, \text{after}, v \rangle$ is an operational semantics of program Π (satisfying hypotheses (7) to (19)).

4.1. *Interpretation of Assertions*

4.1.1. *Set-Theoretic Structure for A*

A set-theoretic structure for A is a function M with domain A such that:

(53) For each sort $\mathbf{s}$ in A, $M(\mathbf{s})$ is a non-empty set (of values for objects of sort $\mathbf{s}$). In particular,

- For the sort $\mathbf{b}$ of truth values, $M(\mathbf{b}) = \{tt, ff\}$,
- For the sorts $\mathbf{s} \in \Upsilon$ of program variables, we have $M(\mathbf{s}) = v(\mathbf{s})$ (therefore the set of possible values of program variables of type $\mathbf{s}$ is the same in the logic and in the program semantics, see (18)).

(54) For every relation symbol $r: (\mathbf{s}_1 \times \cdots \times \mathbf{s}_{\#r} \to \mathbf{b})$ in A, $M(r) \in (M(\mathbf{s}_1) \times \cdots \times M(\mathbf{s}_{\#r}) \to \{tt, ff\})$. In particular $M(true) = tt$ and $M(false) = ff$. Moreover, for all $\pi \in \gamma^*[\![\Pi]\!]$, $M(at[\![\pi]\!]) = \text{at}[\![\pi]\!]$, $M(in[\![\pi]\!]) = \text{in}[\![\pi]\!]$ and $M(after[\![\pi]\!]) = \text{after}[\![\pi]\!]$.

(55) For every function symbol $f: (\mathbf{s}_1 \times \cdots \times \mathbf{s}_{\#f} \to \mathbf{s})$, we have $M(f) \in (M(\mathbf{s}_1) \times \cdots \times M(\mathbf{s}_{\#f}) \to M(\mathbf{s}))$. In particular, if c is an individual constant of sort $\mathbf{s}$, $M(c) \in M(\mathbf{s})$.

4.1.2. *Interpretation of Terms*

(56) An *assigment* δ *in* M assigns a meaning $\delta(x)$ belonging to the set $M(\Delta(x))$ to each free logical variable $x \in V_l^f$ of sort $\Delta(x)$.

We can now define, by induction on the complexity of terms, the interpretation of a term t of sort $\mathbf{s}$ as a function $I(t)$ which maps assignments (giving a meaning to logical free variables) and states (giving a meaning to control assertions and program variables) to elements of $M(\mathbf{s})$. More precisely,

(57) For every term $t \in A$ of sort $\mathbf{s}$, assignment δ, and state $s \in S[\![\Pi]\!]$, we define the interpretation $I(t)[\delta, s]$ as follows:

- If $t \in V_l^f$ is a logical free variable then $I(t)[\delta, s] = \delta(t)$,
- If $t \in V_p^f$ is a program free variable then $I(t)[\delta, s] = v(t)(s)$,
- If t is an individual constant then $I(t)[\delta, s] = M(t)$,
- If t is a term $f(t_1, ..., t_n)$ where f is an n-ary function symbol then $I(f(t_1, ..., t_n))[\delta, s] = M(f)(I(t_1)[\delta, s], ..., I(t_n)[\delta, s])$.

4.1.3. *Interpretation of Assertions*

We now define the satisfaction relation:

$$\models_I P[\delta, s]$$

(read: the assignment δ and state s satisfy assertion P in I) for all assignments δ, states $s \in S[\![\Pi]\!]$, and assertions $P \in A$ as follows:

Atomic Assertions ($t_1, ..., t_n$ are terms; r, a relational symbol):

(58) $\models_I (t_1 \equiv t_2)[\delta, s]$ iff $I(t_1)[\delta, s] = I(t_2)[\delta, s]$

(59) $\models_I r(t_1, ..., t_n)[\delta, s]$ iff $M(r)(I(t_1)[\delta, s], ..., I(t_n)[\delta, s])$ is tt

(60) $\models_I at[\![\pi]\!][\delta, s]$ iff $\mathrm{at}[\![\pi]\!](s)$ is tt

(61) $\models_I in[\![\pi]\!][\delta, s]$ iff $\mathrm{in}[\![\pi]\!](s)$ is tt

(62) $\models_I after[\![\pi]\!][\delta, s]$ iff $\mathrm{after}[\![\pi]\!](s)$ is tt.

Assertions ($P, Q \in A$, $\theta \subseteq A$, $w \in V_l^b$, $x \in V_l^f$, $u \in V_p^b$, $y \in V_p^f$):

(63) $\models_I \neg P[\delta, s]$ iff not $\models_I P[\delta, s]$

(64) $\models_I (\wedge \theta)[\delta, s]$ iff $\models_I P[\delta, s]$ for all P of θ

(65) $\models_I (\vee \theta)[\delta, s]$ iff $\models_I P[\delta, s]$ for some P of θ

(66) $\models_I (P \Rightarrow Q)[\delta, s]$ iff either not $\models_I P[\delta, s]$ or else $\models_I Q[\delta, s]$

(67) $\models_I (P \equiv Q)[\delta, s]$ iff $\models_I (P \Rightarrow Q)[\delta, s]$ and $\models_I (Q \Rightarrow P)[\delta, s]$.

(In the following we use δ_x^v for the assignment δ' which agrees with δ except that $\delta'(x) = v$.)

(68) $\models_I (\exists w.P(w/x))[\delta, s]$ iff there is a $v \in M(\Delta(x))$ such that $\models_I P[\delta_x^v, s]$

(69) $\models_I (\forall w.P(w/x))[\delta, s]$ iff for all $v \in M(\Delta(x))$, $\models_I P[\delta_x^v, s]$.

(In the following when $s,\ s' \in S[\![\Pi]\!]$ we write $s' \cong s \backslash y$ iff for all $\zeta \in (V_p - \{y\})$, $v[\![\zeta]\!](s') = v[\![\zeta]\!](s)$, and for all $\pi \in \gamma^*[\![\Pi]\!]$ we have $\text{at}[\![\pi]\!](s') = \text{at}[\![\pi]\!](s)$ and $\text{in}[\![\pi]\!](s') = \text{in}[\![\pi]\!](s)$ and $\text{after}[\![\pi]\!](s') = \text{after}[\![\pi]\!](s)$.)

(70) $\models_I (\exists u.P(u/y))[\delta, s]$ iff there is an $s' \in S[\![\Pi]\!]$ such that $s' \cong s \backslash y$ and $\models_I P[\delta, s']$

(71) $\models_I (\forall u.P(u/y))[\delta, s]$ iff for all $s' \in S[\![\Pi]\!]$ such that $s' \cong s \backslash y$ we have $\models_I P[\delta, s']$.

We make a distinction between logical and program variables: logical variables are understood with respect to logical assignments whereas program variables are understood with respect to program states. Contrary to Apt (1981) we do not identify assignments and states because the meaning of logical variables is always static whereas the meaning of program variables may be dynamic, that is, depends upon the control part of the states.

4.2. *Interpretation of Asserted Programs*

For all $P,\ Q \in A,\ \pi \in \gamma^*[\![\Pi]\!]$ and assignment δ we define the satisfaction relation:

(72) $\models_I \{P\}\ \pi\{Q\}[\delta]$ iff

(a) $\forall s, s' \in S[\![\pi]\!] \cdot (\models_I P[\delta, s] \wedge t[\![\pi]\!](s, s')) \Rightarrow ((\text{in}[\![\pi]\!](s') \wedge \models_I P[\delta, s']) \vee (\text{after}[\![\pi]\!](s') \wedge \models_I Q[\delta, s']))$.

$\{P\}\ \pi\{Q\}$ means that if control is anywhere in π (see (20)) and P holds, then executing one step in π will either leave control in π with P true or leave control at an exit point of π with Q true. This also holds when considering any number of steps as shown by the following:

(73) THEOREM. $\models_I \{P\}\ \pi\{Q\}[\delta]$ iff

(b) $\forall s, s' \in S[\![\pi]\!] \cdot (\models_I P[\delta, s] \wedge \text{in}[\![\pi]\!](s) \wedge t[\![\pi]\!]^*(s, s')) \Rightarrow ((\text{in}[\![\pi]\!](s') \wedge \models_I P[\delta, s']) \vee (\text{after}[\![\pi]\!](s') \wedge \models_I Q[\delta, s']))$.

Proof. By definition:

(74) $t[\![\pi]\!]^* = \bigvee_{n \geqslant 0} t[\![\pi]\!]^n$, where

(75) $t[\![\pi]\!]^0(s, s') = (s = s')$

(76) $t[\![\pi]\!]^{n+1}(s, s') = (\exists s'' \in S[\![\pi]\!] \cdot (t[\![\pi]\!]^n(s, s'') \wedge t[\![\pi]\!](s'', s')))$.

Therefore for $n = 1$, (b) obviously implies (a). Reciprocally, according to (74) we just have to prove (by induction on n) that (a) implies

(c) $\forall s, s' \in S[\![\pi]\!] \cdot (\models_I P[\delta, s] \wedge \text{in}[\![\pi]\!](s) \wedge t[\![\pi]\!]^n(s, s')) \Rightarrow ((\text{in}[\![\pi]\!](s') \wedge \models_I P[\delta, s']) \vee (\text{after}[\![\pi]\!](s') \wedge \models_I Q[\delta, s']))$.

The basis $n=0$ obviously follows from (75). For the induction step, assuming (s, s' are universally quantified over $S[\![\pi]\!]$),

(d) $\models_I P[\delta, s] \wedge \mathrm{in}[\![\pi]\!](s) \wedge t[\![\pi]\!]^{n+1}(s, s')$; we derive

(e) $\exists s'' \in S[\![\pi]\!].\models_I P[\delta, s] \wedge \mathrm{in}[\![\pi]\!](s) \wedge t[\![\pi]\!]^n(s, s'') \wedge \mathrm{in}[\![\pi]\!](s'') \wedge t[\![\pi]\!](s'', s')$, by (d), (76), (20);

(f) $\exists s'' \in S[\![\pi]\!] \cdot ((\mathrm{in}[\![\pi]\!](s'') \wedge \models_I P[\delta, s'']) \vee (\mathrm{after}[\![\pi]\!](s'') \wedge \models_I Q[\delta, s''])) \wedge \mathrm{in}[\![\pi]\!](s'') \wedge t[\![\pi]\!](s'', s')$, by induction hypothesis;

(g) $\exists s'' \in S[\![\pi]\!].\models_I P[\delta, s''] \wedge t[\![\pi]\!](s'', s')$, by (f), (14), (20);

(h) $((\mathrm{in}[\![\pi]\!](s') \wedge \models_I P[\delta, s']) \vee (\mathrm{after}[\![\pi]\!](s') \wedge \models_I Q[\delta, s']))$, by (g), (a). ∎

In particular when $P=Q=I$, $\{I\}\pi\{I\}$ means that a step, hence any number of steps of π leaves I invariant.

(77) THEOREM. $\models_I \{I\}\,\pi\{I\}[\delta]$ *iff*

$$\forall s, s' \in S[\![\pi]\!] \cdot (\models_I I[\delta, s] \wedge t[\![\pi]\!](s, s')) \Rightarrow \models_I I[\delta, s'].$$

Proof. (72), (20). ∎

(78) THEOREM. $\models_I \{I\}\,\pi\{I\}[\delta]$ *iff*

$$\forall s, s' \in S[\![\pi]\!] \cdot (\models_I I[\delta, s] \wedge t[\![\pi]\!]^*(s, s')) \Rightarrow \models_I I[\delta, s'].$$

Proof. One essentially has to prove by induction on $n \geqslant 0$ that

$$\forall s, s' \in S[\![\pi]\!] \cdot (\models_I I[\delta, s] \wedge t[\![\pi]\!]^n(s, s')) \Rightarrow \models_I I[\delta, s']. \quad ∎$$

Remark. Instead of choosing (72) as definition of the interpretation of asserted programs in GHL, we could equivalently and following Lamport and Schneider (1984) have chosen (77). Then $\{P\}\pi\{Q\}$ would have been understood as an abbreviation for $\{in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q\}\ \pi\{in[\![\pi]\!] \Rightarrow P \wedge after[\![\pi]\!] \Rightarrow Q\}$ (see (46) and (47)) from which (72) hence (73) would be easy to derive.

(79) When $P \in A$, we define $\models_I P$ as $\models_I P[\delta, s]$ for all assignments δ and states $s \in S[\![\pi]\!]$.

(80) When $A \subseteq A$, we define $\models_I A$ as $\models_I P$ for all $P \in A$.

(81) When $\phi \in F$, we define $\models_I \phi$ as $\models_I \phi[\delta]$ for all assignments δ.

(82) When $A \subseteq A$ and $\phi \in F$, we define $A \models_I \phi$ as $(\models_I A \Rightarrow \models_I \phi)$.

5. SOUNDNESS

Let I be an interpretation of GHL for a program $\Pi \in P$. The purpose of this paragraph is to show that no provable formula of GHL is incorrect with respect to I.

Since we have left τ partly unspecified and given a meta-formal system for GHL, this correctness condition can only be proved under the assumption that the unspecified part of τ is sound and that the particular instances of the meta-axiom schemata are valid.

HYPOTHESES.

(83) *The unspecified part of* τ *is sound. For all* $\pi \in (\gamma^*[\![\Pi]\!] - \alpha[\![\Pi]\!])$, *assignment* δ *and* $s \in S[\![\Pi]\!]$,

(84) $\models_I \mathrm{AT}[\![\pi]\!](loc[\![\pi']\!] \mid \pi' \in \gamma[\![\pi]\!] \vee \pi \in \gamma[\![\pi']\!])[\delta, s]$ *only if* $\mathrm{at}[\![\pi]\!](s)$

(85) $\models_I \mathrm{IN}[\![\pi]\!](loc[\![\pi']\!] \mid \pi' \in \gamma[\![\pi]\!] \vee \pi \in \gamma[\![\pi']\!])[\delta, s]$ *only if* $\mathrm{in}[\![\pi]\!](s)$

(86) $\models_I \mathrm{AFTER}[\![\pi]\!](loc[\![\pi']\!] \mid \pi' \in \gamma[\![\pi]\!] \vee \pi \in \gamma[\![\pi']\!])[\delta, s]$ *only if* $\mathrm{after}[\![\pi]\!](s)$.

(87) With these hypotheses the classical soundness proof for τ can be easily adapted to handle assignments and states which play similar rôles. Moreover, the soundness of (32) follows from (11), (84), (12); the soundness of (33) follows from (13), (85); the soundness of (34) follows from (14), (15), (86).

We now consider the soundness of proof system H. We need hypotheses about the meta-axiom schemata (37).

HYPOTHESES. *For all* $\pi \in \gamma^*[\![\Pi]\!]$ *and* $a \in \alpha[\![\pi]\!]$ *we assume*

(88a) *Either* (37a) $\in H$ *and for all* $Q \in A$ *and assignments* δ,

$$\forall s \in S[\![a]\!] \cdot (\models_I \mathrm{PRE}[\![a]\!](Q)[\delta, s] \Rightarrow (\forall s' \in S[\![a]\!].t[\![a]\!](s, s') \Rightarrow \models_I Q[\delta, s']))$$

(88b) *or* (37b) $\in H$ *and for all* $P \in A$ *and assignments* δ,

$$\forall s' \in S[\![a]\!] \cdot ((\exists s \in S[\![a]\!] \cdot (\models_I P[\delta, s] \wedge t[\![a]\!](s, s'))) \Rightarrow \models_I \mathrm{POST}[\![a]\!](P)[\delta, s']).$$

The soundness of H can be proved independently of the soundness of τ provided all assertions of A which appear in the proof are assumed to be valid:

(89) *Soundness of* H:

$$\forall A \subseteq A,\ P, Q \in A. \qquad A \vdash_H \{P\}\ \pi\ \{Q\} \Rightarrow A \models_I \{P\}\ \pi\ \{Q\}.$$

Proof. One proves, by induction on n that if $\psi_1, \ldots, \psi_n$ is a proof of $\{P\}\ \pi\{Q\}$ from A in H then $\models_I \{P\}\ \pi\{Q\}$. This amounts to the proof that the axiom schemata of H are valid and that the inference rules preserve the truth under I of the asserted programs. Hence by case analysis, we have

(90) $\models_I \{in[\![\pi]\!]\}\ \pi\{true\}$

$\Leftrightarrow \forall \delta. \models_I \{in[\![\pi]\!]\}\ \pi\{true\}[\delta]$, by (81)

$\Leftrightarrow \forall\delta, \forall s, s' \in S[\![\pi]\!]. (\models_I in[\![\pi]\!][\delta, s] \wedge t[\![\pi]\!](s, s'))$
$\Rightarrow ((\text{in}[\![\pi]\!](s') \wedge \models_I in[\![\pi]\!][\delta, s'])$
$\vee (\text{after}[\![\pi]\!](s') \wedge \models_I true[\delta, s'])),$ by (72)

$\Leftrightarrow \forall\delta, \forall s, s' \in S[\![\pi]\!], (\text{in}[\![\pi]\!](s) \wedge t[\![\pi]\!](s, s')) \Rightarrow (\text{in}[\![\pi]\!](s') \vee$ after$[\![\pi]\!](s'))$, by (61), (59), (54)

$\Leftarrow$ (20).

The same way one can prove

(91) $\models_I \{true\}\pi\{after[\![\pi]\!]\}$

(92a) $\models_I \{\text{PRE}[\![a]\!](Q)\}\, a\{Q\}$

$\Leftrightarrow \forall\delta, \forall s, s' \in S[\![a]\!] \cdot (\models_I \text{PRE}[\![a]\!](Q)[\delta, s] \wedge t[\![a]\!](s, s'))$
$\Rightarrow ((\text{in}[\![a]\!](s') \wedge \models_I \text{PRE}[\![a]\!](Q)[\delta, s']) \vee (\text{after}[\![a]\!](s')$
$\wedge \models_I Q[\delta, s'])),$ by (81), (72)

$\Leftrightarrow \forall\delta, \forall s \in S[\![a]\!], \models_I \text{PRE}[\![a]\!](Q)[\delta, s] \Rightarrow (\exists s' \in S[\![a]\!].t[\![a]\!](s, s')$
$\Rightarrow \models_I Q[\delta, s'])$, by (17), (14)

$\Leftarrow$ (88a)

(92b) $\models_I \{P\}\, a\{\text{POST}[\![a]\!](P)\}$

$\Leftrightarrow \forall\delta, \forall s \in S[\![a]\!] \cdot (\exists s' \in S[\![a]\!] \cdot (\models_I P[\delta, s] \wedge t[\![a]\!](s, s'))$
$\Rightarrow \models_I \text{POST}[\![a]\!](P)[\delta, s'])$, by (81), (72), (17), (14)

$\Leftarrow$ (88b).

(93) *Decomposition principle*:

$(\forall\pi' \in \gamma[\![\pi]\!] \cdot \models_I \{I\}\, \pi'\{I\})$
$\Leftrightarrow (\forall\pi' \in \gamma[\![\pi]\!], \forall\delta, \forall s, s' \in S[\![\pi']\!] \cdot (\models_I I[\delta, s] \wedge t[\![\pi']\!](s, s')) \Rightarrow \models_I I[\delta, s'])$, by (81), (77)

$\Leftrightarrow (\forall\pi' \in \gamma[\![\pi]\!], \forall\delta, \forall s, s' \in S[\![\pi']\!], \forall a \in \alpha[\![\pi']\!] \cdot$
$(\models_I I[\delta, s] \wedge t[\![a]\!](s, s')) \Rightarrow \models_I I[\delta, s'])$, by (9)

$\Leftrightarrow (\forall\pi' \in \gamma[\![\pi]\!], \forall\delta, \forall s, s' \in S[\![\pi']\!], \forall a \in \alpha[\![\pi']\!] \cdot$
$(\models_I I[\delta, s] \wedge \text{in}[\![a]\!](s) \wedge t[\![a]\!](s, s') \wedge [\text{in}[\![a]\!](s')$
$\vee \text{after}[\![a]\!](s')] \Rightarrow \models_I I[\delta, s'])$, by (20)

$\Leftrightarrow (\forall\pi' \in \gamma[\![\pi]\!], \forall\delta, \forall a \in \alpha[\![\pi']\!], \forall s, s' \in \{s \in S[\![\pi']\!] \mid \text{in}[\![a]\!](s)$
$\vee \text{after}[\![a]\!](s) \cdot$
$(\models_I I[\delta, s] \wedge \text{in}[\![a]\!](s) \wedge t[\![a]\!](s, s') \wedge [\text{in}[\![a]\!](s')$
$\vee \text{after}[\![a]\!](s')] \Rightarrow \models_I I[\delta, s'])$

$\Leftrightarrow (\forall\pi' \in \gamma[\![\pi]\!], \forall\delta, \forall a \in \alpha[\![\pi']\!], \forall s, s' \in S[\![a]\!] \cdot$
$(\models_I I[\delta, s] \wedge \text{in}[\![a]\!](s) \wedge t[\![a]\!](s, s') \wedge [\text{in}[\![a]\!](s')$
$\vee \text{after}[\![a]\!](s')] \Rightarrow \models_I I[\delta, s'])$, by (16)

$\Leftrightarrow (\forall a \in \alpha[\![\pi]\!], \forall\delta, \forall s, s' \in S[\![a]\!] \cdot$
$(\models_I I[\delta, s) \wedge \text{in}[\![a]\!](s) \wedge t[\![a]\!](s, s') \wedge [\text{in}[\![a]\!](s')$
$\vee \text{after}[\![a]\!](s')] \Rightarrow \models_I I[\delta, s'])$.

Because $\alpha[\![\pi]\!] = \bigcup_{\pi' \in \gamma[\![\pi]\!]} \alpha[\![\pi]\!]$ by (4), (5), (6),

$\Leftrightarrow (\forall \delta, \forall a \in \alpha[\![\pi]\!], \forall s, s' \in \{s \in S[\![\pi]\!] \mid \mathrm{in}[\![a]\!](s) \vee \mathrm{after}[\![a]\!](s)\} \cdot$
$(\models_I I[\delta, s] \wedge \mathrm{in}[\![a]\!](s) \wedge t[\![a]\!](s, s') \wedge [\mathrm{in}[\![a]\!](s')$
$\vee \mathrm{after}[\![a]\!](s')] \Rightarrow \models_I I[\delta, s'])$, by (16), (6)

$\Leftrightarrow (\forall \delta, \forall a \in \alpha[\![\pi]\!], \forall s, s' \in S[\![\pi]\!] \cdot$
$(\models_I I[\delta, s]) \wedge \mathrm{in}[\![a]\!](s) \wedge t[\![a]\!](s, s') \wedge [\mathrm{in}[\![a]\!](s')$
$\vee \mathrm{after}[\![a]\!](s')] \Rightarrow \models_I I[\delta, s'])$

$\Leftrightarrow (\forall \delta, \forall a \in \alpha[\![\pi]\!], \forall s, s' \in S[\![\pi]\!] \cdot (\models_I I[\delta, s) \wedge t[\![a]\!](s, s'))$
$\Rightarrow \models_I I[\delta, s'])$, by (20)

$\Leftrightarrow (\forall \delta, \forall s, s' \in S[\![\pi]\!] \cdot (\models_I I[\delta, s) \wedge t[\![\pi]\!](s, s'))$
$\Rightarrow \models_I I[\delta, s'])$, by (9)

$\Leftrightarrow \models_I \{I\} \pi \{I\}$, by (77), (80).

(94) *Locality rule*:

$(\models_I \{in[\![\pi]\!] \wedge P\} \pi \{after[\![\pi]\!] \wedge Q\})$
$\Leftrightarrow (\forall \delta, \forall s, s' \in S[\![\pi]\!] \cdot (\models_I (in[\![\pi]\!] \wedge P)[\delta, s] \wedge t[\![\pi]\!](s, s'))$
$\Rightarrow ((\mathrm{in}[\![\pi]\!](s') \wedge \models_I (in[\![\pi]\!] \wedge P)[\delta, s'])$
$\vee (\mathrm{after}[\![\pi]\!](s') \wedge \models_I (after[\![\pi]\!] \wedge Q)[\delta, s'])))$, by (81), (72)

$\Leftrightarrow (\forall \delta, \forall s, s' \in S[\![\pi]\!] \cdot (\models_I P[\delta, s] \wedge t[\![\pi]\!](s, s'))$
$\Rightarrow ((\mathrm{in}[\![\pi]\!](s') \wedge \models_I P[\delta, s']) \vee (\mathrm{after}[\![\pi]\!](s')$
$\wedge \models_I Q[\delta, s'])))$, by (64), (61), (62), (20)

$\Leftrightarrow (\models_I \{P\} \pi \{Q\})$, by (72), (81).

(95) *Conjunction rule*:

$\{\forall i \in [1, n], \models_I \{P_i\} \pi \{Q_i\}$
$\Leftrightarrow (\forall i \in [1, n], \forall \delta, \forall s, s' \in S[\![\pi]\!] \cdot (\models_I P_i[\delta, s] \wedge t[\![\pi]\!](s, s'))$
$\Rightarrow ((\mathrm{in}[\![\pi]\!](s') \wedge \models_I P_i[\delta, s]) \vee (\mathrm{after}[\![\pi]\!](s')$
$\wedge \models_I Q_i[\delta, s'])))$, by (81), (72)

$\Rightarrow \forall i \in [1, n], \forall \delta, \forall s, s' \in S[\![\pi]\!] \cdot$

$$\left(\left(\models_I \left(\bigwedge_{j=1}^{n} P_j\right)[\delta, s] \wedge t[\![\pi]\!](s, s') \wedge \mathrm{in}[\![\pi]\!](s')\right) \Rightarrow \models_I P_i[\delta, s']\right)$$

$$\wedge \left(\left(\models_I \left(\bigwedge_{j=1}^{n} P_j\right)[\delta, s] \wedge t[\![\pi]\!](s, s') \wedge \mathrm{after}[\![\pi]\!](s')\right) \Rightarrow \models_I Q_i[\delta, s']\right),$$

by (14), (64)

$\Leftrightarrow \forall \delta, \forall s, s' \in S[\![\pi]\!] \cdot$

$$\left(\left(\models_I \left(\bigwedge_{j=1}^{n} P_j\right)[\delta, s] \wedge t[\![\pi]\!](s,s') \wedge \mathrm{in}[\![\pi]\!](s')\right) \Rightarrow \models_I \left(\bigwedge_{i=1}^{n} P_i\right)[\delta, s']\right)$$

$$\wedge \left(\left(\models_I \left(\bigwedge_{j=1}^{n} P_j \right) [\delta, s] \wedge t[\![\pi]\!](s, s') \wedge \text{after}[\![\pi]\!](s') \right) \right.$$

$$\left. \Rightarrow \models_I \left(\bigwedge_{i=1}^{n} Q_i \right) [\delta, s'] \right), \text{ by (64)}$$

$$\Leftrightarrow \forall \delta, \forall s, s' \in S[\![\pi]\!] \cdot$$

$$\left(\models_I \left(\bigwedge_{j=1}^{n} P_j \right) [\delta, s] \wedge t[\![\pi]\!](s, s') \right)$$

$$\Rightarrow \left(\left(\text{in}[\![\pi]\!](s') \wedge \models_I \left(\bigwedge_{i=1}^{n} P_i \right) [\delta, s'] \right) \vee \left(\text{after}[\![\pi]\!](s') \right. \right.$$

$$\left. \left. \wedge \models_I \left(\bigwedge_{i=1}^{n} Q_i \right) [\delta, s] \right) \right), \text{ by (14)}$$

$$\Leftrightarrow \models_I \left\{ \bigwedge_{i=1}^{n} P_i \right\} \pi \left\{ \bigwedge_{i=1}^{n} Q_i \right\}, \text{ by (72), (81).}$$

(96) *Right consequence rule*:

$$(\models_I \{P\}\, \pi \{Q\} \wedge (Q \Rightarrow R) \in A)$$
$$\Rightarrow (\models_I \{P\}\, \pi \{Q\} \wedge \models_I (Q \Rightarrow R)), \text{ since } \models_I A \text{ by hypothesis and (80)}$$
$$\Leftrightarrow (\forall \delta, \forall s, s' \in S[\![\pi]\!] \cdot (\models_I P[\delta, s] \wedge t[\![\pi]\!](s, s'))$$
$$\Rightarrow ((\text{in}[\![\pi]\!](s') \wedge \models_I P[\delta, s']) \vee (\text{after}[\![\pi]\!] \wedge \models_I Q[\delta, s'])))$$
$$\wedge (\forall \delta, \forall s \in S[\![\Pi]\!], (\models_I Q[\delta, s]) \Rightarrow (\models_I R[\delta, s])),$$
by (81), (72), (79), (66), (16)
$$\Rightarrow (\forall \delta, \forall s, s' \in S[\![\pi]\!] \cdot (\models_I P[\delta, s] \wedge t[\![\pi]\!](s, s'))$$
$$\Rightarrow ((\text{in}[\![\pi]\!](s') \wedge \models_I P[\delta, s']) \vee (\text{after}[\![\pi]\!] \wedge \models_I R[\delta, s'])))$$
$$\Leftrightarrow (\models_I \{P\}\, \pi \{R\}), \text{ by (81), (72).}$$

(97) *Left identity rule*: The soundness proof is similar to the above one (using (67) instead of (66)).

(98) *Left consequence rule*:

$$((R \Rightarrow P) \in A \wedge \models_I \{P\}\, a \{Q\})$$
$$\Rightarrow (\models_I (R \Rightarrow P) \wedge \models_I \{P\}\, a \{Q\}), \text{ since } \models_I A \text{ by hypothesis and (80)}$$
$$\Leftrightarrow (\forall \delta, \forall s \in S[\![\Pi]\!] \cdot (\models_I R[\delta, s]) \Rightarrow (\models_I P[\delta, s]))$$
$$\wedge (\forall \delta, \forall s, s' \in S[\![a]\!] \cdot (\models_I P[\delta, s] \wedge t[\![a]\!](s, s'))$$
$$\Rightarrow ((\text{in}[\![a]\!](s') \wedge \models_I P[\delta, s]) \vee (\text{after}[\![a]\!](s') \wedge \models_I Q[\delta, s'])))$$
by (79), (66), (72)

$$\Leftrightarrow (\forall \delta, \forall s \in S[\![\Pi]\!] \cdot (\models_I R[\delta, s]) \Rightarrow (\models_I P[\delta, s]))$$
$$\wedge (\forall \delta, \forall s, s' \in S[\![a]\!]. (\models_I P[\delta, s) \wedge t[\![a]\!](s, s'))$$
$$\Rightarrow (\text{after}[\![a]\!](s') \wedge \models_I Q[\delta, s'])), \text{ by (17), (14)}$$

$$\Rightarrow \forall \delta, \forall s, s' \in S[\![a]\!] \cdot (\models_I R[\delta, s] \wedge t[\![a]\!](s, s'))$$
$$\Rightarrow (\text{after}[\![a]\!](s') \wedge \models_I Q[\delta, s'])$$

$$\Leftrightarrow \forall \delta, \forall s, s' \in S[\![a]\!] \cdot (\models_I R[\delta, s] \wedge t[\![a]\!](s, s'))$$
$$\Rightarrow ((\text{in}[\![a]\!](s') \wedge \models_I R[\delta, s]) \vee (\text{after}[\![a]\!](s') \wedge \models_I Q[\delta, s'])),$$
by (17), (14)

$\Leftrightarrow \models_I \{R\}\, \pi \{Q\}$, by (72), (81). ∎

6. Relative Completeness

Let I be an interpretation of GHL for a program $\Pi \in P$, $\pi \in \gamma^*[\![\Pi]\!]$, and $P, Q \in A$. Assume we have $\models_I \{P\}\, \pi \{Q\}$. We would like to have this provable in $\tau \cup H$, i.e., $\models_{\tau \cup H} \{P\}\, \pi \{Q\}$.

In fact, this cannot be proved in GHL without considering a particular programming language and the corresponding instances of the meta-axiom schemata (37a) or (37b) for all atomic actions or else without assuming the completeness of these meta-axiom schemata.

Hypotheses. *For all* $\pi \in \gamma^*[\![\Pi]\!]$ *and* $a \in \alpha[\![\pi]\!]$ *we assume*

(99a) *Either* (37a) $\in H$ *and for all* $Q \in A$ *and assignments* δ,

$$\forall s \in S[\![a]\!] \cdot ((\forall s' \in S[\![a]\!] \cdot t[\![a]\!](s, s') \Rightarrow \models_I Q[\delta, s']) \Rightarrow \models_I \text{PRE}[\![a]\!](Q)[\delta, s])$$

(99b) *or* (37b) $\in H$ *and for all* $P \in A$ *and assignments* δ,

$$\forall s' \in S[\![a]\!] \cdot (\models_I \text{POST}[\![a]\!](P)[\delta, s'] \Rightarrow (\exists s \in S[\![a]\!] \cdot \models_I P[\delta, s] \wedge t[\![a]\!](s, s'))).$$

Notice that in the case of Hoare's (1969) logic hypothesis (99a) is satisfied (since, e.g., for assignment commands PRE is the *weakest* pre-condition). However, there are two other causes of incompleteness:

(a) Because of the consequence rule, one has to rely upon τ which, by Kurt Gödel's second incompleteness theorem, cannot be consistent, contain arithmetic, and be complete. For this reason, the best one might hope for would be to prove relative completeness of H, that is,

$$\models_I \{P\}\, \pi \{Q\} \Rightarrow \{R \in A \mid \models_I R\} \vdash_H \{P\}\, \pi \{Q\}.$$

(b) Unfortunately in the case of Hoare's logic even this cannot be proved. This is because the necessary intermediate assertions (more

precisely loop invariant assertions) may not be expressible in the assertion language A (Wand, 1978).

Fortunately this is not the case in GHL because whenever $\models_I \{P\}\pi\{Q\}$ holds, P is invariant for π (72), hence P is invariant for loops in π (73). Consequently, and contrary to Hoare's Logic, no loop invariant assertion (stronger than P) is needed in the relative completeness proof (which can be carried out without assuming Cook's expressiveness condition (Cook, 1978)):

(100) $\models_I \{I\}\,\pi\{I\} \Rightarrow \{R \in A \mid \models_I R\} \vdash_H \{I\}\,\pi\{I\}$.

Proof.

$\models_I \{I\}\,\pi\{I\}$

$\Leftrightarrow (\forall \delta, \forall s, s' \in S[\![\pi]\!] \cdot (\models_I I[\delta, s] \wedge t[\![\pi]\!](s, s')) \Rightarrow \models_I I[\delta, s'])$, by (77)

$\Leftrightarrow (\forall \delta, \forall a \in \alpha[\![\pi]\!], \forall s, s' \in \{s \in S[\![\pi]\!] \mid \text{in}[\![a]\!](s) \vee \text{after}[\![a]\!](s)\} \cdot$
$(\models_I I[\delta, s] \wedge \text{in}[\![a]\!](s) \wedge t[\![a]\!](s, s') \wedge \text{after}[\![a]\!](s')) \Rightarrow \models_I I[\delta, s'])$,
by (9), (17), (11)

$\Leftrightarrow (\forall a \in \alpha[\![\pi]\!].$

(β) $\forall \delta, \forall s, s' \in S[\![a]\!] \cdot (\models_I I[\delta, s] \wedge t[\![a]\!](s, s')) \Rightarrow \models_I I[\delta, s'])$,
by (16), (17), (11).

Either (37a) $\in H$ and then

$(\beta) \Leftrightarrow (\forall \delta, \forall s \in S[\![a]\!] \cdot (\models_I I[\delta, s] \Rightarrow (\forall s' \in S[\![a]\!] \cdot t[\![a]\!](s, s') \Rightarrow \models_I I[\delta, s'])))$
$\Rightarrow (\forall \delta, \forall s \in S[\![a]\!] \cdot \models_I I[\delta, s] \Rightarrow \models_I \text{PRE}[\![a]\!](I)[\delta, s])$, by (99a)
$\Leftrightarrow (\models_I (I \Rightarrow \text{PRE}[\![a]\!](I)))$, by (66), (79)

$\Leftrightarrow$ (a) $((I \Rightarrow \text{PRE}[\![a]\!](I)) \in \{R \in A \mid \models_I R\})$.

Or (37b) $\in H$ and then

$(\beta) \Leftrightarrow (\forall \delta, \forall s' \in S[\![a]\!] \cdot (\exists s \in S[\![a]\!] \cdot \models_I I[\delta, s] \wedge t[\![a]\!](s, s')) \Rightarrow \models_I I[\delta, s'])$
$\Rightarrow (\forall \delta, \forall s' \in S[\![a]\!] \cdot \text{POST}[\![a]\!](I)[\delta, s'] \Rightarrow \models_I I[\delta, s'])$, by (99b)
$\Leftrightarrow \vdash_I (\text{POST}[\![a]\!](I) \Rightarrow I)$, by (66), (79)

$\Leftrightarrow$ (b) $((\text{POST}[\![a]\!](I) \Rightarrow I) \in \{R \in A \mid \models_I R\})$.

We can now give the relative completeness proof. $\alpha[\![\pi]\!]$ is finite and for all $a \in \alpha[\![\pi]\!]$ we have either

(c) $\{\text{PRE}[\![a]\!](I)\}\,a\{I\}$, by (37a)

(d) $\{I\}a\{I\}$, by (a), (c), (43)

or

(e) $\{I\}a\{\text{POST}[\![a]\!](I)\}$, by (37b)

(d) $\{I\}a\{I\}$, by (b), (d), (41)

(f) $\alpha[\![\pi]\!] = \bigcup_{a \in \alpha[\![\pi]\!]} \alpha[\![a]\!]$, since $\alpha[\![a]\!] = \{a\}$, by (5), (6)

(g) $\{I\}\pi\{I\}$, by (f), (d), (49). ∎

(101) RELATIVE COMPLETENESS THEOREM.

$$\models_I \{P\}\ \pi\{Q\} \Rightarrow \{R \in A \mid \models_I R\} \vdash_H \{P\}\ \pi\{Q\}.$$

Proof.

$\models_I \{P\}\ \pi\{Q\}$

$\Leftrightarrow (\forall \delta, \forall s, s' \in S[\![\pi]\!] \cdot (\models_I P[\delta, s] \wedge t[\![\pi]\!](s, s'))$
$\Rightarrow ((\text{in}[\![\pi]\!](s') \wedge \models_I P[\delta, s']) \vee (\text{after}[\![\pi]\!](s') \wedge \models_I Q[\delta, s'])))$,
by (81), (72)

$\Leftrightarrow (\forall \delta, \forall s, s' \in S[\![\pi]\!] \cdot (\models_I ((in[\![\pi]\!] \Rightarrow P) \wedge (after[\![\pi]\!] \Rightarrow Q))[\delta, s]$
$\wedge\ t[\![\pi]\!](s, s'))$
$\Rightarrow \models_I ((in[\![\pi]\!] \Rightarrow P) \wedge (after[\![\pi]\!] \Rightarrow Q))[\delta, s'])$,
by (20), (14), (61), (62), (66), (64)

$\Leftrightarrow \models_I \{((in[\![\pi]\!] \Rightarrow P) \wedge (after[\![\pi]\!] \Rightarrow Q))\}\ \pi\{((in[\![\pi]\!] \Rightarrow P)$
$\wedge\ (after[\![\pi]\!] \Rightarrow Q))\}$ (77)

$\Rightarrow \{R \in A \mid \models_I R\} \vdash_H \{((in[\![\pi]\!] \Rightarrow P) \wedge (after[\![\pi]\!] \Rightarrow Q))\}$
$\pi\{((in[\![\pi]\!] \Rightarrow P) \wedge (after[\![\pi]\!] \Rightarrow Q))\}$, by (100)

$\Rightarrow \{R \in A \mid \models_I R\} \vdash_H \{P\}\ \pi\{Q\}$, by (47). ∎

One should not conclude from the fact that relative completeness (101) has been proved without assuming an expressiveness condition à la Cook that GHL is complete in a stronger sense than Hoare's (1969) logic. More precisely, if using GHL we had to prove $P\{\pi\}Q$ in the sense of Hoare, we would have to find a program invariant I such that:

$$(at[\![\pi]\!] \wedge P) \Rightarrow I$$

and

$$\{I\}\ \pi\{Q\}$$

but now, as it is the case for Hoare's logic, no such I might belong to A (except when, but this is not necessary, A is expressive relative to I and P in the sense of Cook (1978).

7. CONCLUSION

We have shown that GHL is a meta-formal system for proving invariance properties of programs which (contrary to Hoare's logic) can be developed (to a large extent) in a programming language independent way. This interesting property allowed us to prove soundness and relative completeness without having to consider a particular language (as in Apt, 1981) for Hoare's logic).

As shown by the relative completeness proof, the central idea of GHL is to use a single global program invariant, which has to be shown to be left

invariant by each atomic action of the program. In most other proof methods, this global invariant would have to be decomposed into local invariants attached to particular program points (or particular values of the control state). The essential difference between these other proof methods is only that different decompositions are used (Cousot, 1980). The advantage of GHL, as shown by Lamport and Schneider (1984), is that, using control predicates, these local invariants can be factored into a global invariant so that the corresponding proof method can be explained in terms of GHL. The disadvantage of GHL is that no decomposition of the global invariant is enforced so that no guideline is offered to the programmer for expressing properties of his program in a simple way.

ACKNOWLEDGMENTS

P. Cousot would like to acknowledge a discussion with F. Schneider on the possible interpretations or mis-interpretations of asserted programs in GHL. We wish to thank W. P. de Roever for helpful comments.

RECEIVED February 16, 1983; ACCEPTED March 7, 1984

REFERENCES

APT, K. R. (1981), Ten years of Hoare's logic, A survey, Part I, *ACM TOPLAS* **3**, No. 4, 431–483.

APT, K. R. (1982), Ten years of Hoare's logic, A survey, Part II, *Theoret. Comput. Sci.* **28**, 83–109.

ASHCROFT, E. A. (1975), Proving properties about parallel programs, *J. Comput. System Sci.* **10**, 110–135.

COOK, S. A. (1978), Soundness and completeness of an axiom system for program verification, *SIAM J. Comput.* **7**, 1, 70–90.

COUSOT, P., AND COUSOT, R. (1980), Reasoning about program invariance proof methods, *in* "Proceedings, Internat. Workshop on Program Construction, Vol. 1, INRIA, Château de Bonas, France.

FEFERMAN, S. (1974), Applications of many-sorted interpolation theorems, *in* "Proceedings, Tarski Symp., pp. 19–32, Amer. Math. Soc., Providence, RI.

FLOYD, R. W. (1967), Assigning meanings to programs, *in* "Proceedings, Symp. on Applied Math.," pp. 19–32, Amer. Math. Soc., Providence, RI.

HOARE, C. A. R. (1969), An axiomatic basis for computer programming, *Comm. ACM* **12**, No. 10, 576–580, 583.

LAMPORT, L. (1980), The "Hoare Logic" of concurrent programs, *Acta Inform.* **14**, 21–37.

LAMPORT, L., AND SCHNEIDER, F. B. (1984), The "Hoare Logic" of CSP, and all that, *ACM TOPLAS* **6**, No. 2, 281–296.

WAND, M. (1978), A new incompleteness result for Hoare's system, *Assoc. Comput. Mach.* **25**, No. 1, 168–175.

Printed in Belgium